"Descobri uma demonstração maravilhosa desta proposição que, no entanto, não cabe nas margens deste ~~livro~~ apresentação" (Pierre de Fermat, Aritmética de Diophanti).

Convergência

Paulo Ricardo Lisboa de Almeida

DSBD

UFPR

Atualização dos pesos

Das aulas passadas, a atualização dos vetores de peso na iteração $\tau + 1$ é dada por.

$$\mathbf{w}^{\tau+1} = \mathbf{w}^{\tau} - \eta \nabla E(\mathbf{w}^{\tau})$$

Onde η é o fator de aprendizagem.

Fator de Aprendizagem

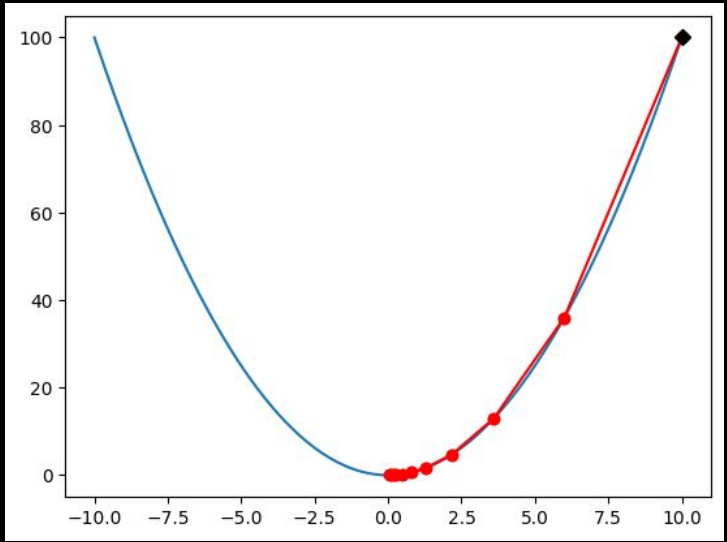
A intuição é que quanto maior o fator de aprendizagem η, mais rápido vamos convergir para a resposta.

Teste essa intuição com o programa disponibilizado no Google Colab.

Fator de Aprendizagem

Encontrando o mínimo da função x^2 considerando x=10 como chute inicial, e:

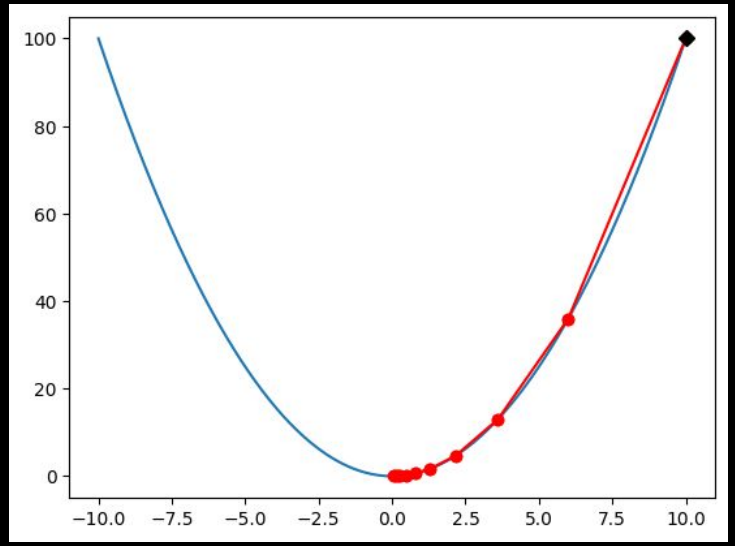
$\eta = 0.2$



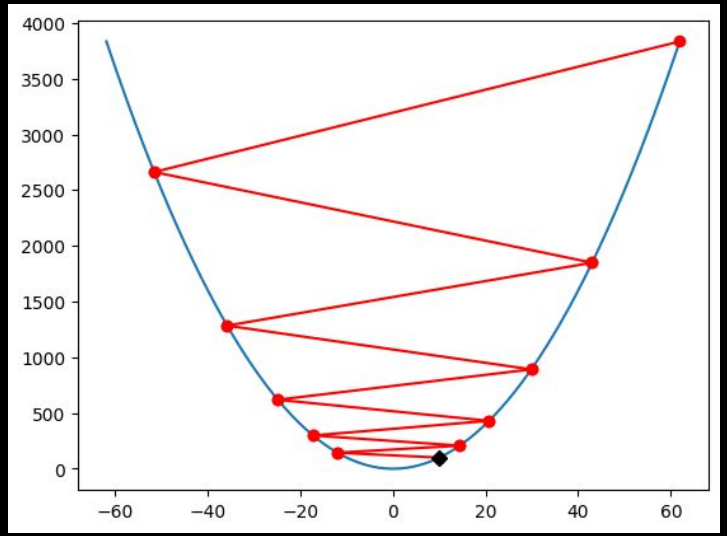
Fator de Aprendizagem

Encontrando o mínimo da função x^2 considerando x=10 como chute inicial, e:

$\eta = 0.2$



$\eta = 1.1$



Fator de Aprendizagem

A derivada de segunda ordem do *loss* em função dos pesos dá o fator de aprendizagem ótimo. Assumindo que a função é aproximadamente quadrática:

$$\eta_{opt} = \left(\frac{\partial^2 L}{\partial w^2}\right)^{-1}$$

Fator de Aprendizagem

A derivada de segunda ordem do *loss* em função dos pesos dá o fator de aprendizagem ótimo. Assumindo que a função é aproximadamente quadrática:

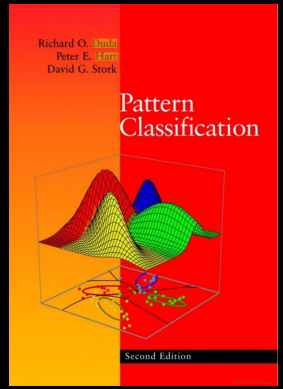
$$\eta_{opt} = \left(\frac{\partial^2 L}{\partial w^2}\right)^{-1}$$

Se $0 < \eta \leq \eta_{opt}$ a convergência é garantida.
Mas pode ser lenta, se $\eta < \eta_{opt}$.

Se $\eta_{opt} < \eta < 2\eta_{opt}$ o sistema oscila.

Se $\eta > 2\eta_{opt}$ o sistema diverge.

Duda, R. O., Hart, P. E., Stork, D. G. Pattern Classification. 2012.



Exemplo

Para a função f(x) = x^2 do exemplo:

$$f(x) = x^2$$

$$f'(x) = 2xdx$$

$$f''(x) = 2dx$$

Logo:

$$\eta_{opt} = (2)^{-1} = \frac{1}{2}$$

Exemplo

Para a função f(x) = x^2 do exemplo:

$$f(x) = x^2$$

$$f'(x) = 2xdx$$

$$f''(x) = 2dx$$

Logo:

$$\eta_{opt} = (2)^{-1} = \frac{1}{2}$$

Valores acima de 0.5 vão fazer o sistema oscilar, e acima de 1 vão fazer o sistema divergir.

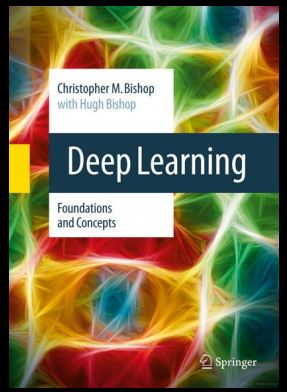
Fator de Aprendizagem

Na prática, apenas garantimos que η seja pequeno o suficiente para garantir convergência.

O cálculo do valor ótimo não é trivial em altas dimensões.

Depende dos autovalores das matrizes.

Veja uma discussão em Bishop, Bishop (2023).



Momentum

Ideia: adicionar o conceito de inércia a descida de gradiente.

Se estávamos “descendo” em determinada direção, tendemos a continuar nessa direção na próxima iteração.

$$\mathbf{w}^{\tau+1} = \mathbf{w}^{\tau} - \eta \nabla E(\mathbf{w}^{\tau}) + \mu \Delta \mathbf{w}^{\tau-1}$$

Onde $0 \leq \mu < 1$.

Momentum

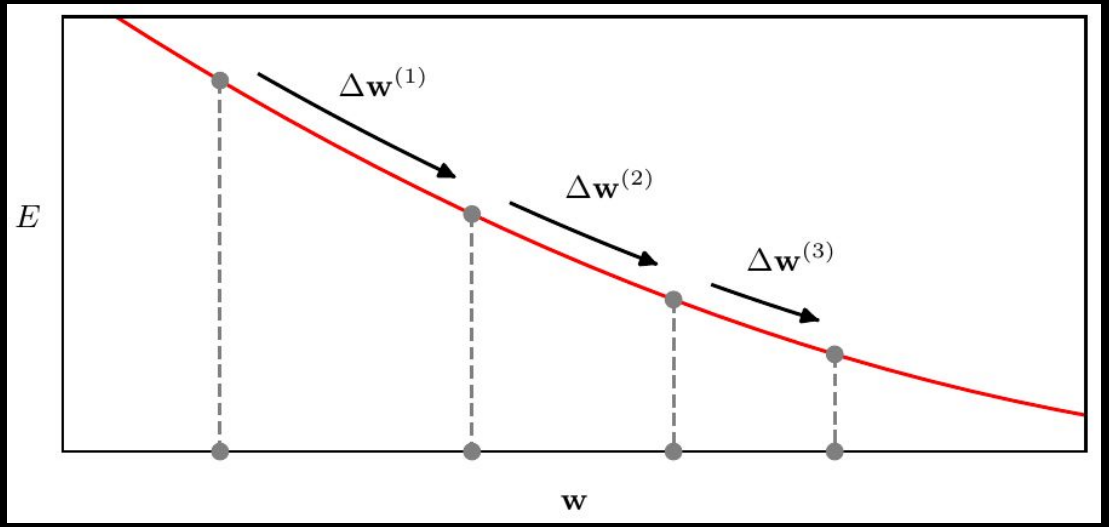
$$\mathbf{w}^{\tau+1} = \mathbf{w}^{\tau} - \eta\nabla E(\mathbf{w}^{\tau}) + \mu\Delta\mathbf{w}^{\tau-1}$$

Gradiente atual.

Atualização do vetor da etapa anterior.

Momentum - Exemplo

$$\mathbf{w}^{\tau+1} = \mathbf{w}^{\tau} - \eta \nabla E(\mathbf{w}^{\tau}) + \mu \Delta \mathbf{w}^{\tau-1}$$



Exemplo de Bishop e Bishop (2024). Na figura **não** é aplicado momentum.

Considere:

- Uma superfície de erro com curvatura relativamente baixa.
- O gradiente ∇E é aproximadamente constante a cada iteração.

Momentum - Exemplo

$$\mathbf{w}^{\tau+1} = \mathbf{w}^{\tau} - \eta \nabla E(\mathbf{w}^{\tau}) + \mu \Delta \mathbf{w}^{\tau-1}$$

Considere:

- Uma superfície de erro com curvatura relativamente baixa.
- O gradiente ∇E é aproximadamente constante a cada iteração.

Momentum - Exemplo

$$\mathbf{w}^{\tau+1} = \mathbf{w}^{\tau} - \eta\nabla E(\mathbf{w}^{\tau}) + \mu\Delta\mathbf{w}^{\tau-1}$$
$$\mathbf{w}^{\tau+1} = \mathbf{w}^{\tau} - \eta\nabla E + \mu(-\eta\nabla E + \mu\Delta\mathbf{w}^{\tau-2})$$
$$\mathbf{w}^{\tau+1} = \mathbf{w}^{\tau} - \eta\nabla E - \mu\eta\nabla E + \mu^2\Delta\mathbf{w}^{\tau-2}$$

Considere:

- Uma superfície de erro com curvatura relativamente baixa.
- O gradiente ∇E é aproximadamente constante a cada iteração.

Momentum - Exemplo

$$\mathbf{w}^{\tau+1} = \mathbf{w}^{\tau} - \eta\nabla E(\mathbf{w}^{\tau}) + \mu\Delta\mathbf{w}^{\tau-1}$$
$$\mathbf{w}^{\tau+1} = \mathbf{w}^{\tau} - \eta\nabla E + \mu(-\eta\nabla E + \mu\Delta\mathbf{w}^{\tau-2})$$
$$\mathbf{w}^{\tau+1} = \mathbf{w}^{\tau} - \eta\nabla E - \mu\eta\nabla E + \mu^2\Delta\mathbf{w}^{\tau-2}$$
$$\mathbf{w}^{\tau+1} = \mathbf{w}^{\tau} - \eta\nabla E - \mu\eta\nabla E + \mu^2(-\eta\nabla E + \mu\mathbf{w}^{\tau-3})$$
$$\mathbf{w}^{\tau+1} = \mathbf{w}^{\tau} - \eta\nabla E - \mu\eta\nabla E - \mu^2\eta\nabla E + \mu^3\mathbf{w}^{\tau-3}$$
...

Considere:

- Uma superfície de erro com curvatura relativamente baixa.
- O gradiente ∇E é aproximadamente constante a cada iteração.

Momentum - Exemplo

$$\mathbf{w}^{\tau+1} = \mathbf{w}^{\tau} - \eta\nabla E(\mathbf{w}^{\tau}) + \mu\Delta\mathbf{w}^{\tau-1}$$

$$\mathbf{w}^{\tau+1} = \mathbf{w}^{\tau} - \eta\nabla E + \mu(-\eta\nabla E + \mu\Delta\mathbf{w}^{\tau-2})$$

$$\mathbf{w}^{\tau+1} = \mathbf{w}^{\tau} - \eta\nabla E - \mu\eta\nabla E + \mu^2\Delta\mathbf{w}^{\tau-2}$$

$$\mathbf{w}^{\tau+1} = \mathbf{w}^{\tau} - \eta\nabla E - \mu\eta\nabla E + \mu^2(-\eta\nabla E + \mu\mathbf{w}^{\tau-3})$$

$$\mathbf{w}^{\tau+1} = \mathbf{w}^{\tau} - \eta\nabla E - \mu\eta\nabla E - \mu^2\eta\nabla E + \mu^3\mathbf{w}^{\tau-3}$$

...

$$\mathbf{w}^{\tau+1} = \mathbf{w}^{\tau} - \eta\nabla E(1 + \mu + \mu^2 + \mu^3 + \dots)$$

$$\mathbf{w}^{\tau+1} = \mathbf{w}^{\tau} - \frac{\eta}{1-\mu}\nabla E$$

Considere:

- Uma superfície de erro com curvatura relativamente baixa.
- O gradiente ∇E é aproximadamente constante a cada iteração.

Momentum - Exemplo

$$\mathbf{w}^{\tau+1} = \mathbf{w}^{\tau} - \eta\nabla E(\mathbf{w}^{\tau}) + \mu\Delta\mathbf{w}^{\tau-1}$$
$$\mathbf{w}^{\tau+1} = \mathbf{w}^{\tau} - \eta\nabla E + \mu(-\eta\nabla E + \mu\Delta\mathbf{w}^{\tau-2})$$
$$\mathbf{w}^{\tau+1} = \mathbf{w}^{\tau} - \eta\nabla E - \mu\eta\nabla E + \mu^2\Delta\mathbf{w}^{\tau-2}$$
$$\mathbf{w}^{\tau+1} = \mathbf{w}^{\tau} - \eta\nabla E - \mu\eta\nabla E + \mu^2(-\eta\nabla E + \mu\mathbf{w}^{\tau-3})$$
$$\mathbf{w}^{\tau+1} = \mathbf{w}^{\tau} - \eta\nabla E - \mu\eta\nabla E - \mu^2\eta\nabla E + \mu^3\mathbf{w}^{\tau-3}$$
$$\dots$$
$$\mathbf{w}^{\tau+1} = \mathbf{w}^{\tau} - \eta\nabla E(1 + \mu + \mu^2 + \mu^3 + \dots)$$
$$\mathbf{w}^{\tau+1} = \mathbf{w}^{\tau} - \frac{\eta}{1-\mu}\nabla E$$

Soma de uma série geométrica. Veja um esboço da prova nos últimos slides.

Considere:

- Uma superfície de erro com curvatura relativamente baixa.
- O gradiente ∇E é aproximadamente constante a cada iteração.

Momentum - Exemplo

Com momentum.

$$\mathbf{w}^{\tau+1} = \mathbf{w}^{\tau} - \frac{\eta}{1-\mu}\nabla E$$

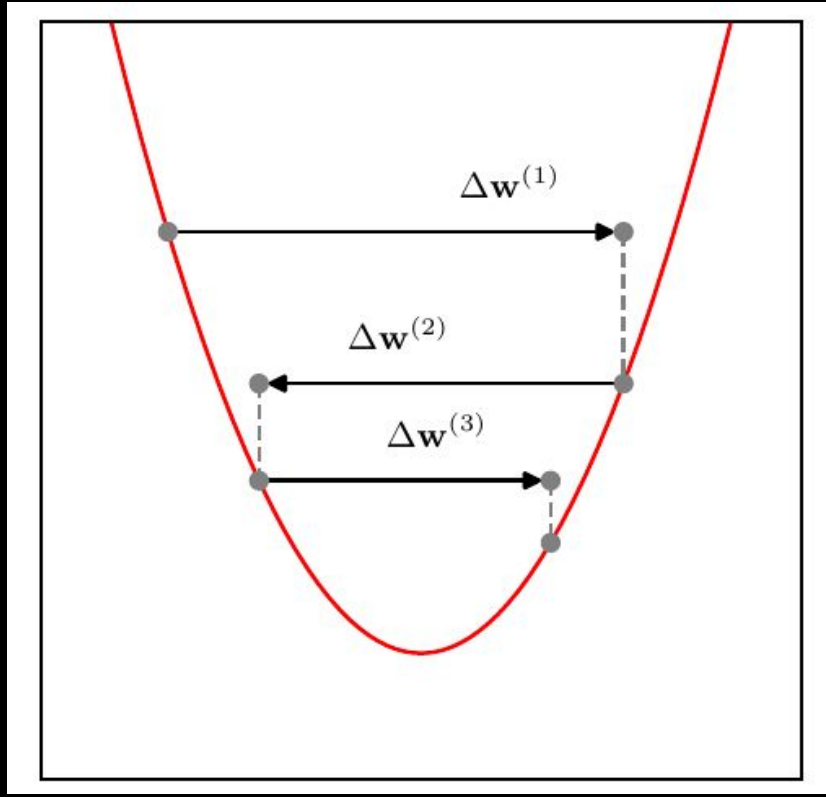
Sem momentum.

$$\mathbf{w}^{\tau+1} = \mathbf{w}^{\tau} - \eta\nabla E$$

Aceleramos a convergência de η para $\frac{\eta}{1-\mu}$.

Momentum Exemplo 2

E para um caso onde há oscilação, o que esperamos que acontecerá ao adicionar momentum?



Exemplo de Bishop e Bishop (2024).

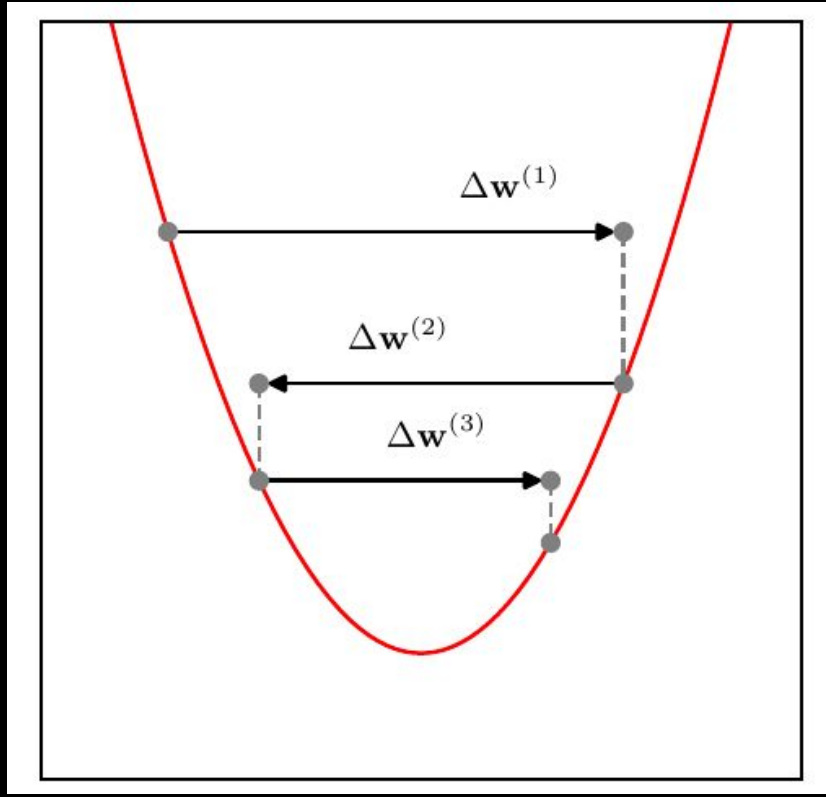
Momentum Exemplo 2

E para um caso onde há oscilação, o que esperamos que acontecerá ao adicionar momentum?

Os termos do momentum tendem a se cancelar.

Horas temos $+\nabla E$, e em outras temos $-\nabla E$.

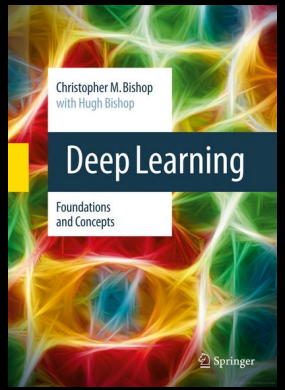
Logo, o *learning rate* será aproximadamente o mesmo que se não tivéssemos momentum: η



Exemplo de Bishop e Bishop (2024).

Na prática

Na prática, um valor comum é $\mu = 0.9$.



Bishop, Bishop (2024).

Learning rate scheduler

Intuição: no começo do treinamento, um *learning rate* maior pode levar a uma convergência mais rápida, e fugir de mínimos locais.

> Ao custo de causar oscilações próximo dos pontos de mínimo, onde um *learning rate* pequeno possibilitaria um melhor ajuste fino.

Learning rate scheduler

Intuição: no começo do treinamento, um *learning rate* maior pode levar a uma convergência mais rápida, e fugir de mínimos locais.

> Ao custo de causar oscilações próximo dos pontos de mínimo, onde um *learning rate* pequeno possibilitaria um melhor ajuste fino.

Para tentar alcançar o melhor dos dois mundos, podemos usar um *Learning Rate Scheduler* (Escalonador de Fator de Aprendizado).

Escalonadores

Alguns exemplos de escalonadores:

- Linear.
- Lei da potência.
- Decaimento exponencial.

Exemplo - Escalonador Linear

$$\eta^{\tau} = \begin{cases} (1 - \frac{\tau}{K})\eta^{0} + \frac{\tau}{K}\eta^{K} & , se\ \tau \leq K \\ \eta^{K} & , se\ \tau > K \end{cases}$$

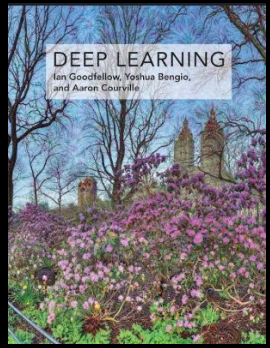
O learning rate inicia em η^{0} e decai linearmente até η^{K} durante *K* passos. Após isso, é mantido fixo em η^{K}.

Exemplo - Escalonador Linear

É difícil escolher os valores de K, η^0 e η^K.

Segundo Goodfellow, Bengio e Courville, $\eta^K = 0.01\eta^0$ pode ser um bom chute inicial.

Goodfellow, I., Bengio, Y., Courville, A. Deep Learning. 2016.



De qualquer forma, analisar empiricamente a curva de aprendizado pode nos indicar bons valores para essas constantes.

$$\eta^\tau = \begin{cases} (1 - \frac{\tau}{K})\eta^0 + \frac{\tau}{K}\eta^K & , se\ \tau \leq K \\ \eta^K & , se\ \tau > K \end{cases}$$

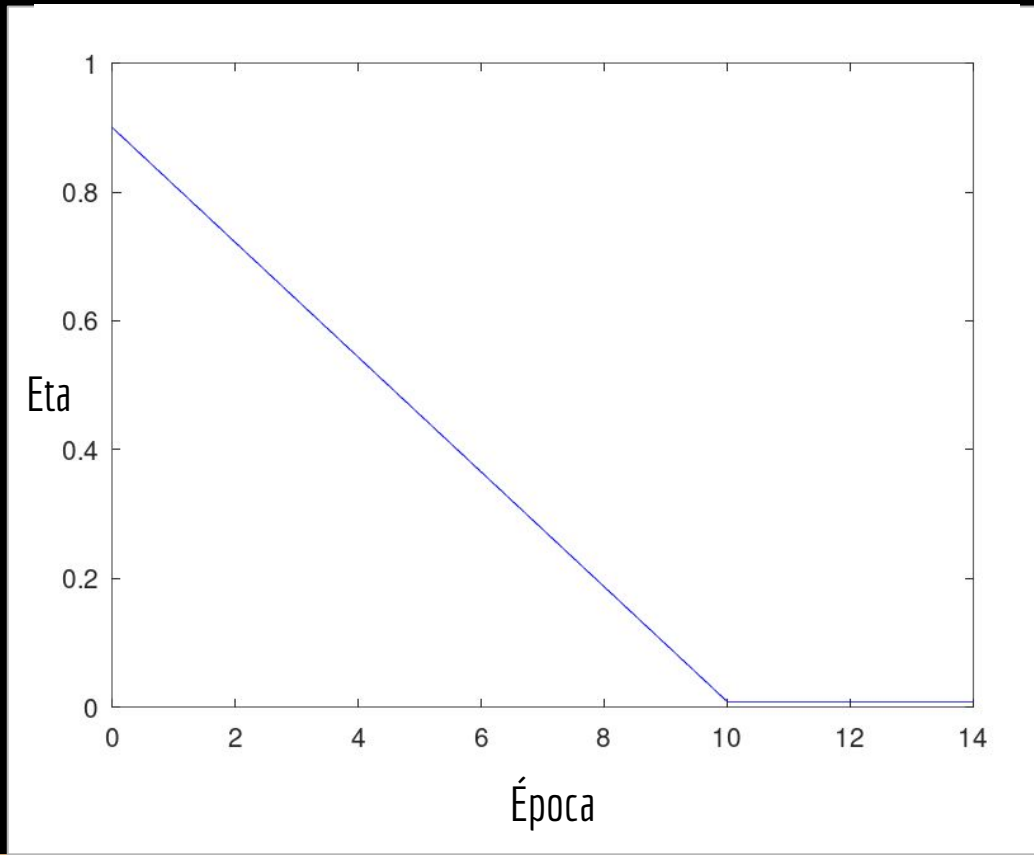
Exemplo - Escalonador Linear

Exemplo considerando:

η^0 = 0.9

η^K = 0.009

K = 10

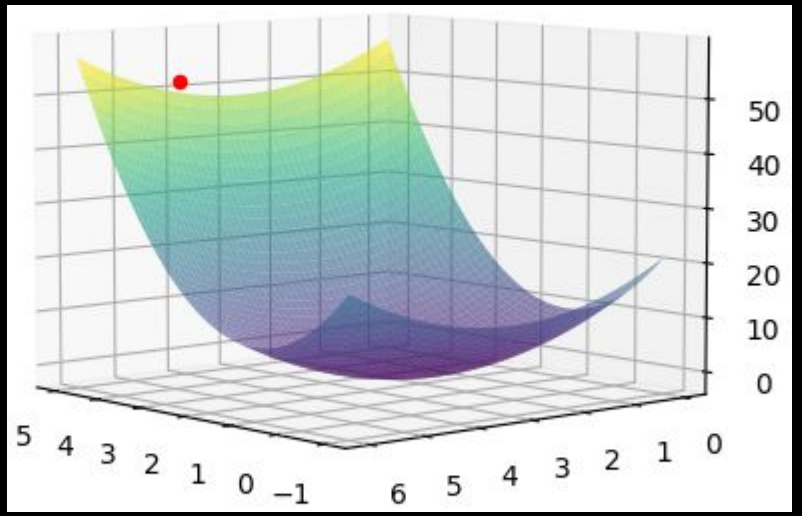


RMSProp

O momentum tenta otimizar o *learning rate* de forma global.

Mas a curvatura do erro geralmente varia dependendo da direção (ex.: entre o eixo x e y).

Ideia: ter um learning *rate diferente* para cada parâmetro.

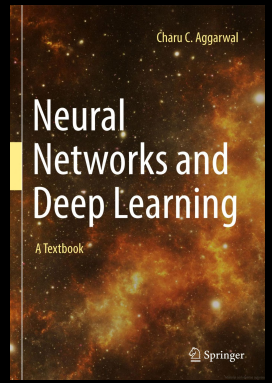


RMSProp

O RMSProp usa uma ideia similar ao Momentum, com as diferenças:

- É calculado para cada um dos parâmetros.
- Faz a raiz da soma dos quadrados dos gradientes (para detalhes, pesquise sobre o AdaGrad).

Aggarwal. Neural Networks and Deep Learning. 2018.



RMSProp

$$r_i^{\tau+1} = \beta r_i^{\tau} + (1-\beta)\left(\frac{\partial E(\mathbf{w})}{\partial w_i}\right)^2$$

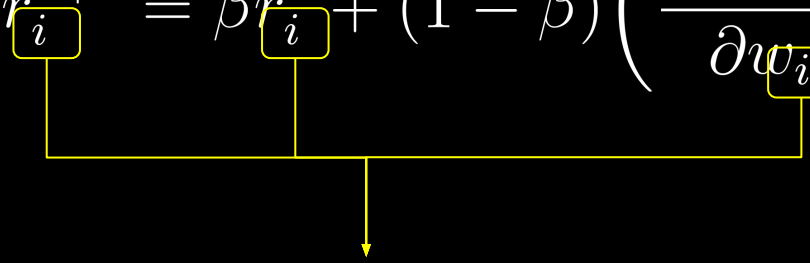
$$0 < \beta < 1$$

$$w_i^{\tau+1} = w_i^{\tau} - \frac{\eta}{\sqrt{r_i^{\tau+1}} + \delta}\left(\frac{\partial E(\mathbf{w})}{\partial w_i}\right)$$

RMSProp

$$r_i^{\tau+1} = \beta r_i^{\tau} + (1-\beta)\left(\frac{\partial E(\mathbf{w})}{\partial w_i}\right)^2$$

$$0 < \beta < 1$$



Calculado para o i-ésimo parâmetro.

$$w_i^{\tau+1} = w_i^{\tau} - \frac{\eta}{\sqrt{r_i^{\tau+1}} + \delta}\left(\frac{\partial E(\mathbf{w})}{\partial w_i}\right)$$

RMSProp

$$r_i^{\tau+1} = \beta r_i^{\tau} + (1-\beta)\left(\frac{\partial E(\mathbf{w})}{\partial w_i}\right)^2$$

$$0 < \beta < 1$$

O último gradiente terá peso $(1-\beta)$, o penúltimo $(1-\beta)^2$, ... decaindo exponencialmente.

$$w_i^{\tau+1} = w_i^{\tau} - \frac{\eta}{\sqrt{r_i^{\tau+1}} + \delta}\left(\frac{\partial E(\mathbf{w})}{\partial w_i}\right)$$

RMSProp

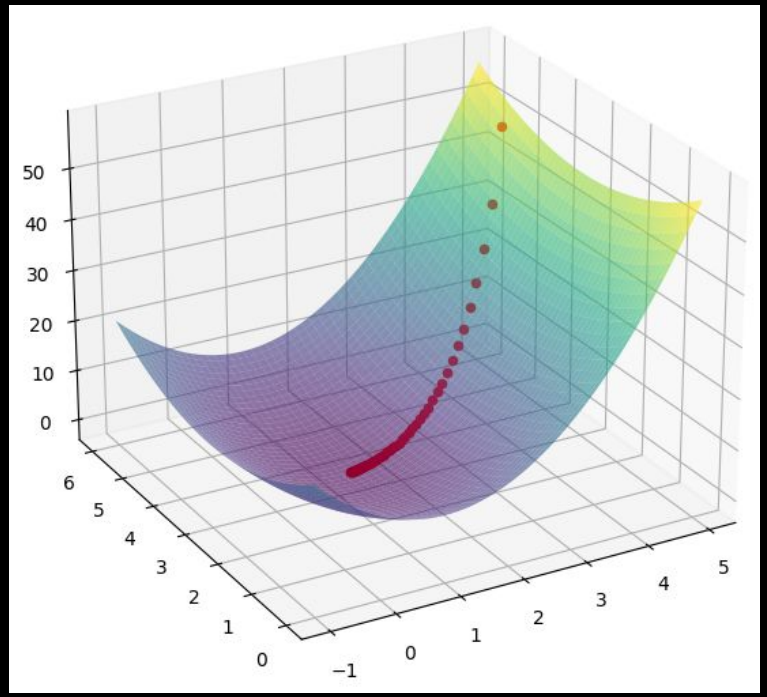
$$r_i^{\tau+1} = \beta r_i^{\tau} + (1-\beta)\left(\frac{\partial E(\mathbf{w})}{\partial w_i}\right)^2 \qquad 0 < \beta < 1$$

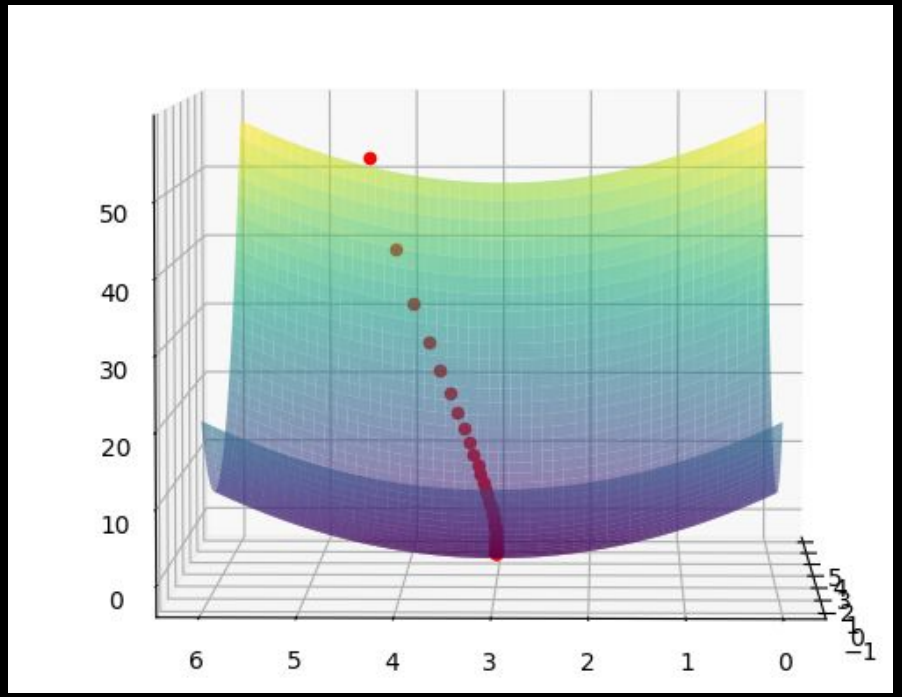
$$w_i^{\tau+1} = w_i^{\tau} - \frac{\eta}{\sqrt{r_i^{\tau+1}} + \delta}\left(\frac{\partial E(\mathbf{w})}{\partial w_i}\right)$$

Constante para garantir estabilidade numérica caso $r_i^{\tau+1}$ fique próximo de zero. Geralmente $\delta = 10^{-8}$.

Faça você mesmo

Entenda e execute o exemplo do RMSProp disponibilizado no Google Colab.

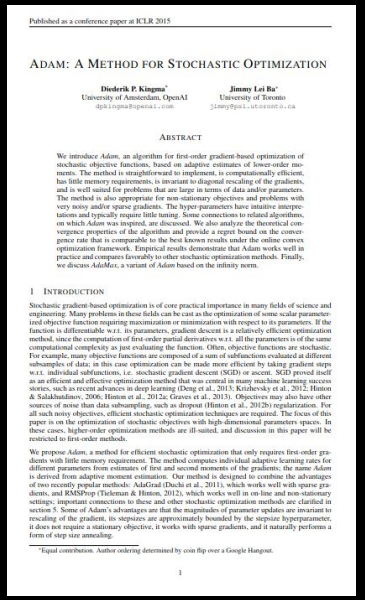




Adam

O algoritmo Adam é um dos mais famosos e usados para ajustar o learning rate.

Utiliza uma combinação de RMSProp e Momentum.



Published as a conference paper at ICLR 2015

ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION

Diederik P. Kingma*
University of Amsterdam, OpenAI
dpkingma@openai.com

Jimmy Lei Ba*
University of Toronto
jimmy@psi.utoronto.ca

ABSTRACT

We introduce *Adam*, an algorithm for first-order gradient-based optimization of stochastic objective functions, based on adaptive estimates of lower-order moments. The method is straightforward to implement, is computationally efficient, has little memory requirements, is invariant to diagonal rescaling of the gradients, and is well suited for problems that are large in terms of data and/or parameters. The method is also appropriate for non-stationary objectives and problems with very noisy and/or sparse gradients. The hyper-parameters have intuitive interpretations and typically require little tuning. Some connections to related algorithms, on which *Adam* was inspired, are discussed. We also analyze the theoretical convergence properties of the algorithm and provide a regret bound on the convergence rate that is comparable to the best known results under the online convex optimization framework. Empirical results demonstrate that Adam works well in practice and compares favorably to other stochastic optimization methods. Finally, we discuss *AdaMax*, a variant of *Adam* based on the infinity norm.

1 INTRODUCTION

Stochastic gradient-based optimization is of core practical importance in many fields of science and engineering. Many problems in these fields can be cast as the optimization of some scalar parameterized objective function requiring maximization or minimization with respect to its parameters. If the function is differentiable w.r.t. its parameters, gradient descent is a relatively efficient optimization method, since the computation of first-order partial derivatives w.r.t. all the parameters is of the same computational complexity as just evaluating the function. Often, objective functions are stochastic. For example, many objective functions are composed of a sum of subfunctions evaluated at different subsamples of data; in this case optimization can be made more efficient by taking gradient steps w.r.t. individual subfunctions, i.e. stochastic gradient descent (SGD) or ascent. SGD proved itself as an efficient and effective optimization method that was central in many machine learning success stories, such as recent advances in deep learning (Deng et al., 2013; Krizhevsky et al., 2012; Hinton & Salakhutdinov, 2006; Hinton et al., 2012a; Graves et al., 2013). Objectives may also have other sources of noise than data subsampling, such as dropout (Hinton et al., 2012b) regularization. For all such noisy objectives, efficient stochastic optimization techniques are required. The focus of this paper is on the optimization of stochastic objectives with high-dimensional parameters spaces. In these cases, higher-order optimization methods are ill-suited, and discussion in this paper will be restricted to first-order methods.

We propose *Adam*, a method for efficient stochastic optimization that only requires first-order gradients with little memory requirement. The method computes individual adaptive learning rates for different parameters from estimates of first and second moments of the gradients; the name *Adam* is derived from adaptive moment estimation. Our method is designed to combine the advantages of two recently popular methods: AdaGrad (Duchi et al., 2011), which works well with sparse gradients, and RMSProp (Tieleman & Hinton, 2012), which works well in on-line and non-stationary settings; important connections to these and other stochastic optimization methods are clarified in section 5. Some of Adam's advantages are that the magnitudes of parameter updates are invariant to rescaling of the gradient, its stepsizes are approximately bounded by the stepsize hyperparameter, it does not require a stationary objective, it works with sparse gradients, and it naturally performs a form of step size annealing.

*Equal contribution. Author ordering determined by coin flip over a Google Hangout.

Kingma, Diederik P., and Jimmy Ba. "Adam: A method for stochastic optimization." *arXiv preprint arXiv:1412.6980* (2014).

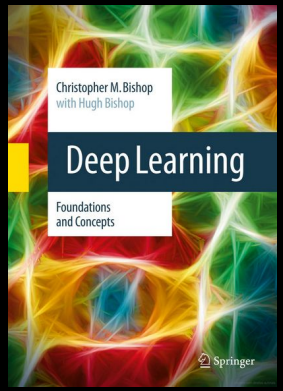
Adam

$$s_i^{\tau+1} = \beta_1 s_i^{\tau} + (1-\beta_1)\left(\frac{\partial E(\mathbf{w})}{\partial w_i}\right)$$

$$r_i^{\tau+1} = \beta_2 r_i^{\tau} + (1-\beta_2)\left(\frac{\partial E(\mathbf{w})}{\partial w_i}\right)^2$$

$$w_i^{\tau+1} = w_i^{\tau} - \eta \frac{s_i^{\tau+1}}{\sqrt{r_i^{\tau+1}} + \delta}$$

Aqui foi omitida a correção do Bias.
Veja em Bishop e Bishop (2024).



Adam

$$s_i^{\tau+1} = \beta_1 s_i^{\tau} + (1-\beta_1)\left(\frac{\partial E(\mathbf{w})}{\partial w_i}\right)$$

← Momentum, com decaimento controlado por β_1.

$$r_i^{\tau+1} = \beta_2 r_i^{\tau} + (1-\beta_2)\left(\frac{\partial E(\mathbf{w})}{\partial w_i}\right)^2$$

← Momentum, com decaimento controlado por β_2.

$$w_i^{\tau+1} = w_i^{\tau} - \eta \frac{s_i^{\tau+1}}{\sqrt{r_i^{\tau+1}} + \delta}$$

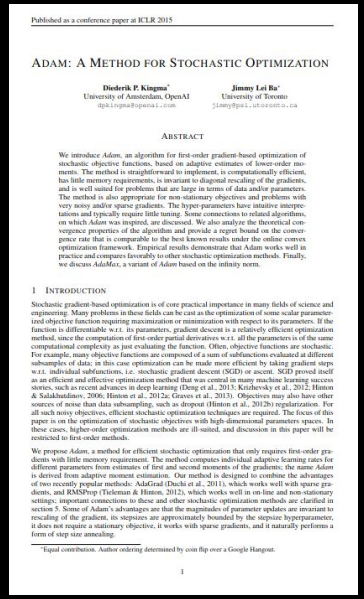
Adam

$$s_i^{\tau+1} = \beta_1 s_i^{\tau} + (1-\beta_1)\left(\frac{\partial E(\mathbf{w})}{\partial w_i}\right)$$

$$r_i^{\tau+1} = \beta_2 r_i^{\tau} + (1-\beta_2)\left(\frac{\partial E(\mathbf{w})}{\partial w_i}\right)^2$$

$$w_i^{\tau+1} = w_i^{\tau} - \eta \frac{s_i^{\tau+1}}{\sqrt{r_i^{\tau+1}} + \delta}$$

No artigo original é sugerido usar $\beta_1 = 0.9$ e $\beta_2 = 0.99$.



Published as a conference paper at ICLR 2015

ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION

Diederik P. Kingma*
University of Amsterdam, OpenAI
dpkingma@openai.com

Jimmy Lei Ba*
University of Toronto
jimmy@psi.utoronto.ca

ABSTRACT

We introduce *Adam*, an algorithm for first-order gradient-based optimization of stochastic objective functions, based on adaptive estimates of lower-order moments. The method is straightforward to implement, is computationally efficient, has little memory requirements, is invariant to diagonal rescaling of the gradients, and is well suited for problems that are large in terms of data and/or parameters. The method is also appropriate for non-stationary objectives and problems with very noisy and/or sparse gradients. The hyper-parameters have intuitive interpretations and typically require little tuning. Some connections to related algorithms, on which *Adam* was inspired, are discussed. We also analyze the theoretical convergence properties of the algorithm and provide a regret bound on the convergence rate that is comparable to the best known results under the online convex optimization framework. Empirical results demonstrate that Adam works well in practice and compares favorably to other stochastic optimization methods. Finally, we discuss *AdaMax*, a variant of *Adam* based on the infinity norm.

1 INTRODUCTION

Stochastic gradient-based optimization is of core practical importance in many fields of science and engineering. Many problems in these fields can be cast as the optimization of some scalar parameterized objective function requiring maximization or minimization with respect to its parameters. If the function is differentiable w.r.t. its parameters, gradient descent is a relatively efficient optimization method, since the computation of first-order partial derivatives w.r.t. all the parameters is of the same computational complexity as just evaluating the function. Often, objective functions are stochastic. For example, many objective functions are composed of a sum of subfunctions evaluated at different subsamples of data; in this case optimization can be made more efficient by taking gradient steps w.r.t. individual subfunctions, i.e. stochastic gradient descent (SGD) or ascent. SGD proved itself as an efficient and effective optimization method that was central in many machine learning success stories, such as recent advances in deep learning (Deng et al., 2013; Krizhevsky et al., 2012; Hinton & Salakhutdinov, 2006; Hinton et al., 2012a; Graves et al., 2013). Objectives may also have other sources of noise than data subsampling, such as dropout (Hinton et al., 2012b) regularization. For all such noisy objectives, efficient stochastic optimization techniques are required. The focus of this paper is on the optimization of stochastic objectives with high-dimensional parameters spaces. In these cases, higher-order optimization methods are ill-suited, and discussion in this paper will be restricted to first-order methods.

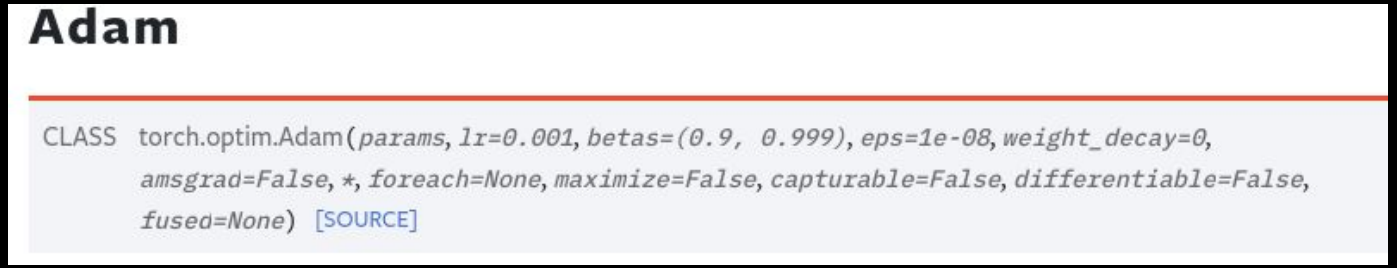
We propose *Adam*, a method for efficient stochastic optimization that only requires first-order gradients with little memory requirement. The method computes individual adaptive learning rates for different parameters from estimates of first and second moments of the gradients; the name *Adam* is derived from adaptive moment estimation. Our method is designed to combine the advantages of two recently popular methods: AdaGrad (Duchi et al., 2011), which works well with sparse gradients, and RMSProp (Tieleman & Hinton, 2012), which works well in on-line and non-stationary settings; important connections to these and other stochastic optimization methods are clarified in section 5. Some of Adam's advantages are that the magnitudes of parameter updates are invariant to rescaling of the gradient, its stepsizes are approximately bounded by the stepsize hyperparameter, it does not require a stationary objective, it works with sparse gradients, and it naturally performs a form of step size annealing.

*Equal contribution. Author ordering determined by coin flip over a Google Hangout.

Kingma, Diederik P., and Jimmy Ba. "Adam: A method for stochastic optimization." *arXiv preprint arXiv:1412.6980* (2014).

Adam - Pytorch

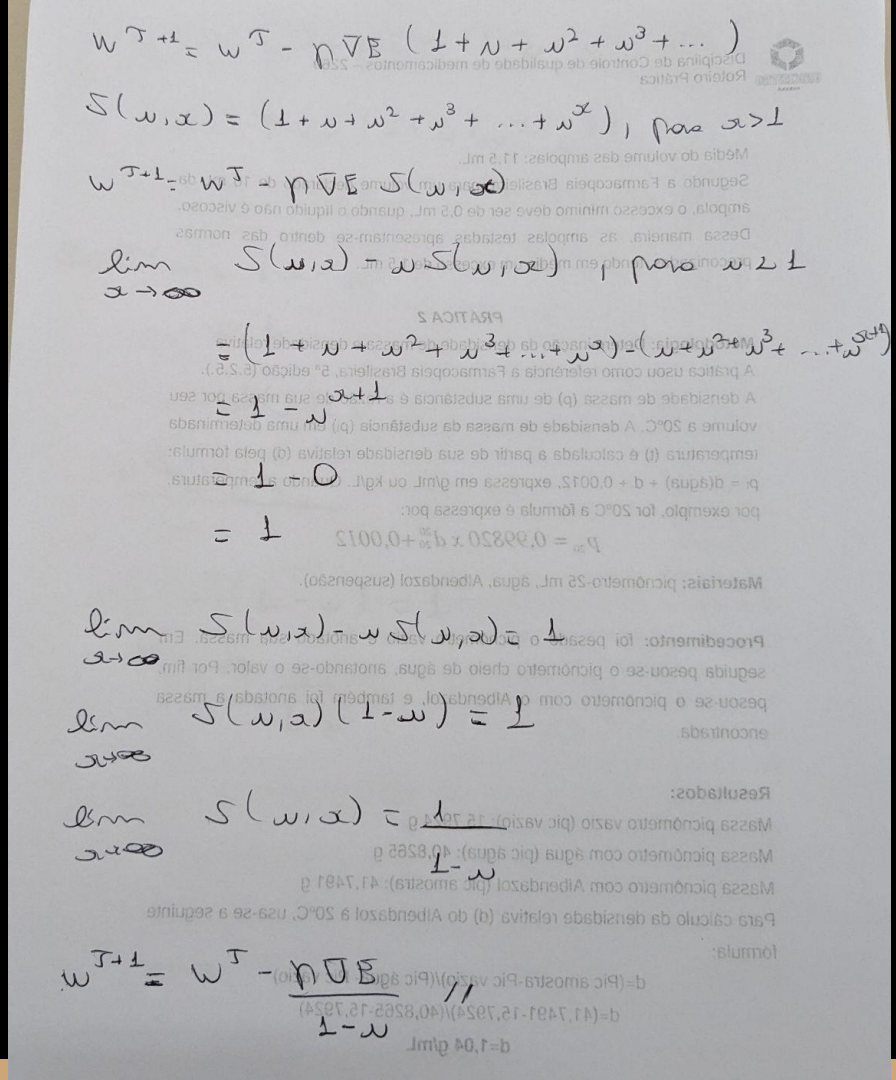
pytorch.org/docs/stable/generated/torch.optim.Adam.html



Exercício

1. Modifique o algoritmo da aula passada, que treinava uma rede neural para classificação de vagas, para utilizar o Adam como otimizador. Compare a convergência da rede usando *learning rate* fixo (ou com adição um scheduler), versus usando Adam (com ou sem scheduler).

Anexo - Esboço da prova



Referências

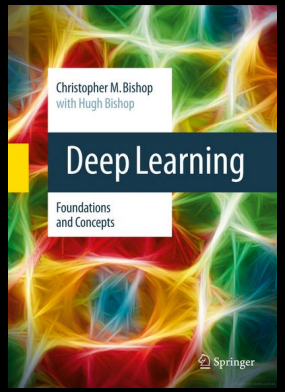
Bishop, C. M., Bishop, H. Deep Learning: Foundations and Concepts. 2024.

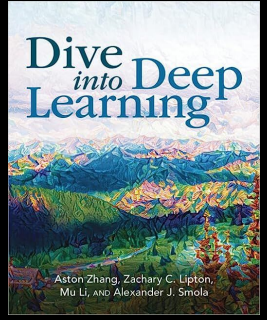
Zhang, A., Lipton, Z. C., Li, M., Smola, A. J. Dive Into Deep Learning. 2023

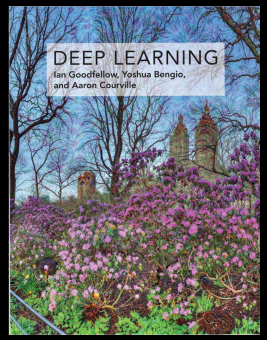
Goodfellow, I., Bengio, Y., Courville, A. Deep Learning. 2016.

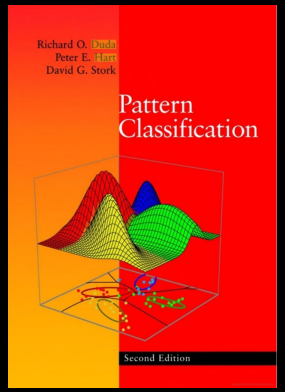
Duda, R. O., Hart, P. E., Stork, D. G. Pattern Classification. 2012.

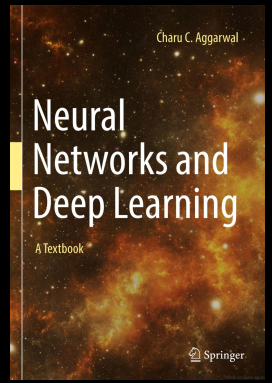
Aggarwal. Neural Networks and Deep Learning. 2018.











Licença